| | Type | Hits | Search Text | DBs | Time Stamp | Comments | Error Definiti on | Errors |
|---|---|---|---|---|---|---|---|---|
| 1 | BRS | 6947 | "virtual machine" | USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB | 2004/04/05 17:27 | | | 0 |
| 2 | BRS | 653 | "start pointer" | USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB | 2004/04/05 17:27 | | | 0 |
| 3 | BRS | 745 | mutex | USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB | 2004/04/05 17:28 | | | 0 |
| 4 | BRS | 107 | "virtual machine" and mutex | USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB | 2004/04/05 17:28 | | | 0 |
| 5 | BRS | 11 | "virtual machine" and "start pointer" and mutex | USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB | 2004/04/05 17:52 | | | 0 |
| 6 | BRS | 5 | "virtual machine" same mutex | USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB | 2004/04/05 17:53 | | | 0 |
| 7 | BRS | 15 | "start pointer" and mutex | USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB | 2004/04/07 15:26 | | | 0 |
| 8 | BRS | 27 | "virtual machine" and "start pointer" | USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB | 2004/04/06 15:02 | | | 0 |
| 9 | IS&R | 56 | (("4675829") or ("4924408") or ("5210876") or ("5301260") or ("5301325") or ("5339436") or ("5367685") or ("5442792") or ("5450575") or ("5452457") or ("5469574") or ("5530964") or ("5551040") or ("5590332") or ("5598561") or ("5603030") or ("5613120") or ("5655122") or ("5675804") or ("5721854") or ("5761513") or ("5764989") or ("5815720") or ("5835771") or ("5848274") or ("5857104") or ("5872978") or ("5873104")).PN. | USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB | 2004/04/06 15:07 | | | 0 |
| 10 | BRS | 19 | ((("4675829") or ("4924408") or ("5210876") or ("5301260") or ("5301325") or ("5339436") or ("5367685") or ("5442792") or ("5450575") or ("5452457") or ("5469574") or ("5530964") or ("5551040") or ("5590332") or ("5598561") or ("5603030") or ("5613120") or ("5655122") or ("5675804") or ("5721854") or ("5761513") or ("5764989") or ("5815720") or ("5835771") or ("5848274") or ("5857104") or ("5872978") or ("5873104")).PN.) and pointer | USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB | 2004/04/06 19:32 | | | 0 |
| 11 | BRS | 50 | read$3 near3 "start pointer" | USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB | 2004/04/06 19:34 | | | 0 |
| 12 | BRS | 4031 | multi-thread$2 | USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB | 2004/04/07 15:25 | | | 0 |
| 13 | BRS | 654 | "start pointer" | USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB | 2004/04/07 15:26 | | | 0 |

|  | Type | Hits | Search Text | DBs | Time Stamp | Comments | Error Definiti on | Errors |
|---|---|---|---|---|---|---|---|---|
| 14 | BRS | 11 | multi-thread$2 same "start pointer" | USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB | 2004/04/07 15:27 |  |  | 0 |
| 15 | BRS | 22 | multi-thread$2 and "start pointer" | USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB | 2004/04/07 15:27 |  |  | 0 |

IEEE HOME | SEARCH IEEE | SHOP | WEB ACCOUNT | CONTACT IEEE

Membership Publications/Services Standards Conferences Careers/Jobs

IEEE Xplore® RELEASE 1.6

Welcome
United States Patent and Trademark Office

IEEE Xplore® 1 Million Documents 1 Million Users And Growing

» Search Results

Help FAQ Terms IEEE Peer Review Quick Links

Welcome to IEEE Xplore®
- Home
- What Can I Access?
- Log-out

Tables of Contents
- Journals & Magazines
- Conference Proceedings
- Standards

Search
- By Author
- Basic
- Advanced

Member Services
- Join IEEE
- Establish IEEE Web Account
- Access the IEEE Member Digital Library

Your search matched **2** of **1022101** documents.
A maximum of **500** results are displayed, **15** to a page, sorted by **Relevance** in **Descending** order.

**Refine This Search:**
You may refine your search by editing the current search expression or entering a new one in the text box.

multi thread and pointer Search

☐ Check to search within this result set

**Results Key:**
**JNL** = Journal or Magazine **CNF** = Conference **STD** = Standard

1 **Pointer cache assisted prefetching**
*Collins, J.; Sair, S.; Calder, B.; Tullsen, D.M.;*
Microarchitecture, 2002. (MICRO-35). Proceedings. 35th Annual IEEE/ACM International Symposium on , 18-22 Nov. 2002
Pages:62 - 73

[Abstract] [PDF Full-Text (271 KB)] **IEEE CNF**

2 **Compile/run-time support for thread migration**
*Hai Jiang; Chaudhary, V.;*
Parallel and Distributed Processing Symposium., Proceedings International, IPDPS 2002, Abstracts and CD-ROM , 15-19 April 2002
Pages:58 - 66

[Abstract] [PDF Full-Text (270 KB)] **IEEE CNF**

Home | Log-out | Journals | Conference Proceedings | Standards | Search by Author | Basic Search | Advanced Search | Join IEEE | Web Account | New this week | OPAC Linking Information | Your Feedback | Technical Support | Email Alerting | No Robots Please | Release Notes | IEEE Online Publications | Help | FAQ| Terms | Back to Top

P RTAL
US Patent & Trademark Office

Subscribe (Full Service) Register (Limited Service, Free) Login

Search: ○ The ACM Digital Library ◉ The Guide

"start pointer"+"multi-threaded"+"next entry" SEARCH

THE GUIDE TO COMPUTING LITERATURE

Feedback Report a problem Satisfaction survey

Terms used start pointer multi threaded next entry

Found 4 of 798,953

Sort results by: relevance

Display results: expanded form

Save results to a Binder
Search Tips
☐ Open results in a new window

Try an Advanced Search
Try this search in The Digital Library

Results 1 - 4 of 4

Relevance scale

1 A new flexible VHDL simulator
Arlet Ottens, Henk Corporaal, Wilco van Hoogstraeten
September 1994 **Proceedings of the conference on European design automation**
Full text available: pdf(541.27 KB) Additional Information: full citation, references, index terms

2 Session 8: systems support for multimedia: Cost-effective streaming server implementation using Hi-tactix
Damien Le Moal, Tadashi Takeuchi, Tadaaki Bandoh
December 2002 **Proceedings of the tenth ACM international conference on Multimedia**
Full text available: pdf(271.85 KB) Additional Information: full citation, abstract, references, index terms

High performance and high quality for continuous media stream delivery needed by streaming server systems cannot be achieved efficiently using general-purpose operating systems, due to the overhead of the I/O mechanism implementation generally used. Special OS combined with powerful hardware can deliver better performance and quality but increases development complexity and deployment costs. The External I/O Engine Architecture adopts a hybrid approach, implementing streaming engines using the s ...

**Keywords**: audio/video streaming, operating system, quicktime, real-time

3 The Stanford FLASH multiprocessor
Jeffrey Kuskin, David Ofelt, Mark Heinrich, John Heinlein, Richard Simoni, K. Gharachorloo, J. Chapin, D. Nakahira, J. Baxter, M. Horowitz, A. Gupta, M. Rosenblum, J. Hennessy
August 1998 **25 years of the international symposia on Computer architecture (selected papers)**
Full text available: pdf(1.48 MB) Additional Information: full citation, references, index terms

4 The Stanford FLASH multiprocessor
J. Kuskin, D. Ofelt, M. Heinrich, J. Heinlein, R. Simoni, K. Gharachorloo, J. Chapin, D. Nakahira, J. Baxter, M. Horowitz, A. Gupta, M. Rosenblum, J. Hennessy
April 1994 **ACM SIGARCH Computer Architecture News , Proceedings of the 21ST annual international symposium on Computer architecture**, Volume 22 Issue 2
Full text available: pdf(1.50 MB) Additional Information: full citation, abstract, references, citings, index terms

The FLASH multiprocessor efficiently integrates support for cache-coherent shared memory and high-performance message passing, while minimizing both hardware and software overhead. Each node in FLASH contains a microprocessor, a portion of the machine's global memory, a port to the interconnection network, an I/O interface, and a custom node controller called MAGIC. The MAGIC chip handles all communication both within the node and among nodes, using hardwired data paths for efficient data moveme ...

Results 1 - 4 of 4

PORTAL

US Patent & Trademark Office

Subscribe (Full Service) Register (Limited Service, Free) Login

Search: ○ The ACM Digital Library ◉ The Guide

"start pointer"+"multi-threaded"+mutex SEARCH

THE GUIDE TO COMPUTING LITERATURE

Feedback Report a problem Satisfaction survey

Terms used **start pointer** **multi threaded** **mutex**

Found **101** of **798,953**

| | |
|---|---|
| Sort results by | relevance |
| Display results | expanded form |

Save results to a Binder
Search Tips
☐ Open results in a new window

Try an Advanced Search
Try this search in The Digital Library

printed

Results 1 - 20 of 101 Result page: [✓ 1] 2 3 4 5 6 next

Relevance scale

1 What is Multi-Threading?
Martin McCarthy
February 1997 **Linux Journal**
Full text available: html(25.02 KB) Additional Information: full citation, abstract, citings, index terms

A primer on multi-threading: the process whereby linux manages several tasks simultaneously

2 Thread-Specific Data and Signal Handling in Multi-Threaded Applications
Martin McCarthy
April 1997 **Linux Journal**
Full text available: html(18.96 KB) Additional Information: full citation, abstract, index terms

This second part of a series on Multi-threading deals with how to use C programs with one of the POSIX packages available for Linux to handle signals and concurrent threads

3 A Study of the Structure and Performance of MMU Handling Software
Vikram P. Joshi, Yousef A. Khalidi, Dock Williams
June 1994 Technical Report, Sun Microsystems, Inc.
Full text available: pdf(54.84 KB) Additional Information: full citation, abstract

Modern operating systems provide a rich set of interfaces for mapping, sharing, and protecting memory. Different memory management unit (MMU) architectures provide different mechanisms for managing memory translations. Since the same OS usually runs on different MMU architectures, a software "hardware address translation" (hat) layer that abstracts the MMU architecture is normally implemented between MMU hardware and the virtual memory system of the OS. In this paper, we study the impact of the ...

4 A study of common pitfalls in simple multi-threaded programs
Sung-Eun Choi, E. Christopher Lewis
March 2000 **ACM SIGCSE Bulletin , Proceedings of the thirty-first SIGCSE technical symposium on Computer science education,** Volume 32 Issue 1
Full text available: pdf(555.98 KB) Additional Information: full citation, abstract, references, citings, index terms

It is generally acknowledged that developing correct multi-threaded codes is difficult, because threads may interact with each other in unpredictable ways. The goal of this work is to discover common multi-threaded programming pitfalls, the knowledge of which will be useful in instructing new programmers and in developing tools to aid in multi-threaded programming. To this end, we study multi-threaded applications written by students from introductory operating systems courses. Although the ...

5 POSIX thread libraries
Felix Garcia, Javier Fernandez
February 2000 **Linux Journal**
Full text available: html(20.36 KB) Additional Information: full citation, abstract, references, index terms

The authors have studied five

6 Introduction to Multi-Threaded Programming
Brian Masney
May 1999 **Linux Journal**
Full text available: html(10.32 KB) Additional Information: full citation, abstract, citings, index terms

A description of thread programming basics for C programmers

7 Performance measurements for multithreaded programs
Minwen Ji, Edward W. Felten, Kai Li
June 1998 **ACM SIGMETRICS Performance Evaluation Review , Proceedings of the 1998 ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems**, Volume 26 Issue 1
Full text available: pdf(1.37 MB) Additional Information: full citation, abstract, references, citings, index terms

Multithreaded programming is an effective way to exploit concurrency, but it is difficult to debug and tune a highly threaded program. This paper describes a performance tool called Tmon for monitoring, analyzing and tuning the performance of multithreaded programs. The performance tool has two novel features: it uses "thread waiting time" as a measure and constructs thread waiting graphs to show thread dependencies and thus performance bottlenecks, and it identifies "semi-busy-waiting" points w ...

8 Perfecting preemption threshold scheduling for object-oriented real-time system design: from the perspective of real-time synchronization
Saehwa Kim, Seongsoo Hong, Tae-Hyung Kim
June 2002 **ACM SIGPLAN Notices , Proceedings of the joint conference on Languages, compilers and tools for embedded systems: software and compilers for embedded systems**, Volume 37 Issue 7
Full text available: pdf(380.92 KB) Additional Information: full citation, abstract, references, index terms

In spite of the proliferation of object-oriented design methodologies in contemporary software development, their application to real-time embedded systems has been limited because of the practitioner's conservative attitude toward handling timing constraints. In fact, this conservative attitude is well-grounded because traditional priority-based scheduling techniques cannot be straightforwardly integrated into them. The automated implementation from the object-oriented real-time designs usually ...

**Keywords**: object-oriented real-time system design, preemption threshold scheduling, priority ceiling protocol, priority inheritance protocols, real-time synchronization

9 Eraser: a dynamic data race detector for multi-threaded programs
Stefan Savage, Michael Burrows, Greg Nelson, Patrick Sobalvarro, Thomas Anderson
October 1997 **ACM SIGOPS Operating Systems Review , Proceedings of the sixteenth ACM symposium on Operating systems principles**, Volume 31 Issue 5
Full text available: pdf(1.51 MB) Additional Information: full citation, references, citings, index terms

10 Debuggable concurrency extensions for standard ML
Andrew P. Tolmach, Andrew W. Appel

December 1991 **ACM SIGPLAN Notices , Proceedings of the 1991 ACM/ONR workshop on Parallel and distributed debugging**, Volume 26 Issue 12
Full text available: pdf(1.22 MB) Additional Information: full citation, references, citings, index terms

11 Embedding Python in Multi-Threaded C/C++ Applications
Ivan Pulleyn
May 2000 **Linux Journal**
Full text available: html(18.08 KB) Additional Information: full citation, abstract, index terms

Python provides a clean intuitive interface to complex, threaded applications.

12 Problems with Pthreads and Ada
Offer Pazy
June 1990 **ACM SIGAda Ada Letters , Proceedings of the fourth international workshop on Real-time Ada issues**, Volume X Issue 9
Full text available: pdf(758.81 KB) Additional Information: full citation, abstract, references, index terms

This position paper discusses issues related to the recent effort to add real-time extensions to the posix (1003.1 |3|) standard, and in particular to the proposal for light-weight tasking in the form of pthreads |2|. It is claimed that this effort is relevant to the Ada community, especially in the real-time domain. While offering great opportunities to improve the performance of multi-tasking, real-time Ada applications, it may become not useful if the semantics of the specifications are no ...

13 A thread-aware debugger with an open interface
Daniel Schulz, Frank Mueller
August 2000 **ACM SIGSOFT Software Engineering Notes , Proceedings of the International Symposium on Software Testing and Analysis**, Volume 25 Issue 5
Full text available: pdf(347.13 KB) Additional Information: full citation, abstract, references, citings, index terms

While threads have become an accepted and standardized model for expressing concurrency and exploiting parallelism for the shared-memory model, debugging threads is still poorly supported. This paper identifies challenges in debugging threads and offers solutions to them. The contributions of this paper are threefold. First, an open interface for debugging as an extension to thread implementations is proposed. Second, extensions for thread-aware debugging are identified and implemented wit ...

**Keywords**: active debugging, concurrency, debugging, open interface, threads

14 An efficient meta-lock for implementing ubiquitous synchronization
Ole Agesen, David Detlefs, Alex Garthwaite, Ross Knippel, Y. S. Ramakrishna, Derek White
October 1999 **ACM SIGPLAN Notices , Proceedings of the 14th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**, Volume 34 Issue 10
Full text available: pdf(2.00 MB) Additional Information: full citation, abstract, references, citings, index terms

Programs written in concurrent object-oriented languages, especially ones that employ thread-safe reusable class libraries, can execute synchronization operations (lock, notify, etc.) at an amazing rate. Unless implemented with utmost care, synchronization can become a performance bottleneck. Furthermore, in languages where every object may have its own monitor, per-object space overhead must be minimized. To address these concerns, we have developed a meta-lock to mediate access to synchro ...

**Keywords**: concurrent threads, object-oriented language implementation, synchronization

15 Language support for lightweight transactions
Tim Harris, Keir Fraser
October 2003 **ACM SIGPLAN Notices , Proceedings of the 18th ACM SIGPLAN conference on Object-oriented programing, systems, languages, and applications**, Volume 38 Issue 11
Full text available: pdf(224.15 KB) Additional Information: full citation, abstract, references, index terms

Concurrent programming is notoriously difficult. Current abstractions are intricate and make it hard to design computer systems that are reliable and scalable. We argue that these problems can be addressed by moving to a declarative style of concurrency control in which programmers directly indicate the safety properties that they require. In our scheme the programmer demarks sections of code which execute within lightweight software-based transactions that commit atomically and exactly once. Th ...

**Keywords**: concurrency, conditional critical regions, non-blocking systems, transactions

16 An Efficient Meta-lock for Implementing Ubiquitous Synchronization
Ole Agesen, David Detlefs, Alex Garthwaite, Ross Knippel, Y.S. Ramakrishna, Derek White
April 1999 Technical Report, Sun Microsystems, Inc.
Full text available: pdf(105.75 KB) Additional Information: full citation, abstract

Programs written in concurrent object-oriented languages, especially ones that employ threadsafe reusable class libraries, can execute synchronization operations (lock, notify, etc.) at an amazing rate. Unless implemented with utmost care, synchronization can become a performance bottleneck. Furthermore, in languages where every object may have its own monitor, per-object space overhead must be minimized. To address these concerns, we have developed a meta-lock to mediate access to synch ...

17 Procs and locks: a portable multiprocessing platform for standard ML of New Jersey
J. Gregory Morrisett, Andrew Tolmach
July 1993 **ACM SIGPLAN Notices , Proceedings of the fourth ACM SIGPLAN symposium on Principles and practice of parallel programming**, Volume 28 Issue 7
Full text available: pdf(976.70 KB) Additional Information: full citation, abstract, references, citings, index terms

We have built a portable platform for running Standard ML of New Jersey programs on multiprocessors. It can be used to implement user-level thread packages for multiprocessors within the ML language with first-class continuations. The platform supports experimentation with different thread scheduling policies and synchronization constructs. it has been used to construct a Modula-3 style thread package and a version of Concurrent ML, and has been ported to three different mu ...

18 Automating real-time multi-threaded application development
C. Riva, M. Krieger
November 1995 **Proceedings of the 1995 conference of the Centre for Advanced Studies on Collaborative research**
Full text available: pdf(89.57 KB) Additional Information: full citation, abstract, references, index terms

Over the past decade, the advances made in VLSI devices have driven the development of cost-effective, symmetric multiprocessor architectures. In turn, operating systems have evolved to take advantage of the true concurrency offered by these architectures. However, because of the lack of adequate development tools, delivery of the intrinsic benefits of these platforms directly to applications has proven elusive. This has been particularly the case for the use of these systems in real-time applic ...

19 A customizable substrate for concurrent languages
Suresh Jagannathan, Jim Philbin
July 1992 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1992 conference on Programming language design and implementation**, Volume 27 Issue 7

Full text available: pdf(1.45 MB) Additional Information: full citation, abstract, references, citings, index terms

We describe an approach to implementing a wide-range of concurrency paradigms in high-level (symbolic) programming languages. The focus of our discussion is STING, a dialect of Scheme, that supports lightweight threads of control and virtual processors as first-class objects. Given the significant degree to which the behavior of these objects may be customized, we can easily express a variety of concurrency paradigms and linguistic structures within a common framework without loss of effici ...

20 A foundation for an efficient multi-threaded scheme system
Suresh Jagannathan, Jim Philbin
January 1992 **ACM SIGPLAN Lisp Pointers , Proceedings of the 1992 ACM conference on LISP and functional programming**, Volume V Issue 1

Full text available: pdf(1.19 MB) Additional Information: full citation, abstract, references, citings, index terms

We have built a parallel dialect of Scheme called STING that differs from its contemporaries in a number of important respects. STING is intended to be used as an operating system substrate for modern parallel programming languages. The basic concurrency management objects is STING are first-class lightweight threads of control and virtual processors (VPs). Unlike high-level concurrency structures, STING threads and VPs are not encumbered by complex synchronization protocols. ...

Results 1 - 20 of 101 Result page: **1** 2 3 4 5 6 next

PORTAL
US Patent & Trademark Office

Subscribe (Full Service) Register (Limited Service, Free) Login

Search: ○ The ACM Digital Library ◉ The Guide

"start pointer"+"multi-threaded"+overwrite SEARCH

THE GUIDE TO COMPUTING LITERATURE

Feedback Report a problem Satisfaction survey

Terms used **start pointer** **multi threaded** **overwrite** Found **60** of **798,953**

Sort results by: relevance
Display results: expanded form

Save results to a Binder
Search Tips
☐ Open results in a new window

Try an Advanced Search
Try this search in The Digital Library

printed

Results 1 - 20 of 60 Result page: **1** 2 3 4 next

Relevance scale

1 GC Points in a Threaded Environment
Ole Agesen
December 1998 Technical Report, Sun Microsystems, Inc.
Full text available: pdf(79.07 KB) Additional Information: full citation, abstract

Many garbage-collected systems, including most that involve a stop-the-world phase, restrict GC to so called GC points. In single-threaded environments, GC points carry no overhead: when a GC must be done, the single thread is already at a GC point. In multi-threaded environments, however, only the thread that triggers the GC by failing an allocation will be at a GC point. Other threads must be rolled forward to their next GC point before the GC can take place. We compare, in the context ...

2 The data locality of work stealing
Umut A. Acar, Guy E. Blelloch, Robert D. Blumofe
July 2000 **Proceedings of the twelfth annual ACM symposium on Parallel algorithms and architectures**
Full text available: pdf(259.32 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper studies the data locality of the work-stealing scheduling algorithm on hardware-controlled shared-memory machines. We present lower and upper bounds on the number of cache misses using work stealing, and introduce a locality-guided work-stealing algorithm along with experimental validation. As a lower bound, we show that there is a family of multi-threaded computations Gn each member of which requires &THgr;(

3 Object combining: A new aggressive optimization for object intensive programs
Ronald Veldema, J. H. Ceriel, F. H. Rutger, E. Henri
November 2002 **Proceedings of the 2002 joint ACM-ISCOPE conference on Java Grande**
Full text available: pdf(99.27 KB) Additional Information: full citation, abstract, references, index terms

Object combining tries to put objects together that have roughly the same life times in order to reduce strain on the memory manager and to reduce the number of pointer indirections during a program's execution. Object combining works by appending the fields of one object to another, allowing allocation and freeing of multiple objects with a single heap (de) allocation. Unlike object *inlining*, which will only optimize objects where one has a (unique) pointer to another, our optimization al ...

**Keywords**: Java, garbage collection, object management

4 A randomized implementation of multiple functional arrays
Tyng-Ruey Chuang

July 1994 **ACM SIGPLAN Lisp Pointers , Proceedings of the 1994 ACM conference on LISP and functional programming**, Volume VII Issue 3

Full text available: pdf(1.28 MB) Additional Information: full citation, abstract, references, citings, index terms

The array update problem in the implementation of a purely functional language is the following: once an array is updated, both the original array and the newly updated one must be preserved to maintain referential transparency. Previous approaches have mainly based on the detection or enforcement of single-threaded accesses to an aggregate, by means of compiler-time analyses or language restrictions. These approaches cannot deal with aggregates which are updated in a multi-threaded manner. ...

5 Concurrent garbage collection using hardware-assisted profiling

Timothy H. Heil, James E. Smith

October 2000 **ACM SIGPLAN Notices , Proceedings of the second international symposium on Memory management**, Volume 36 Issue 1

Full text available: pdf(1.74 MB) Additional Information: full citation, abstract, citings, index terms

In the presence of on-chip multithreading, a Virtual Machine (VM) implementation can readily take advantage of *service threads* for enhancing performance by performing tasks such as profile collection and analysis, dynamic optimization, and garbage collection concurrently with program execution. In this context, a hardware-assisted profiling mechanism is proposed. The *Relational Profiling Architecture* (RPA) is designed from the top down. RPA is based on a relational model similar ...

6 Enhancing software reliability with speculative threads

Jeffrey Oplinger, Monica S. Lam

October 2002 **Tenth international conference on architectural support for programming languages and operating systems on Proceedings of the 10th international conference on architectural support for programming languages and operating systems (ASPLOS-X)**, Volume 37 , 36 , 30 Issue 10 , 5 , 5

Full text available: pdf(1.47 MB) Additional Information: full citation, abstract, references, citings

This paper advocates the use of a monitor-and-recover programming paradigm to enhance the reliability of software, and proposes an architectural design that allows software and hardware to cooperate in making this paradigm more efficient and easier to program.We propose that programmers write monitoring functions assuming simple sequential execution semantics. Our architecture speeds up the computation by executing the monitoring functions speculatively in parallel with the main computation. For ...

7 SafetyNet: improving the availability of shared memory multiprocessors with global checkpoint/recovery

Daniel J. Sorin, Milo M. K. Martin, Mark D. Hill, David A. Wood

May 2002 **ACM SIGARCH Computer Architecture News**, Volume 30 Issue 2

Full text available: pdf(1.28 MB) Publisher Site Additional Information: full citation, abstract, references, citings, index terms

We develop an availability solution, called *SafetyNet,* that uses a unified, lightweight checkpoint/recovery mechanism to support multiple long-latency fault detection schemes. At an abstract level, *SafetyNet* logically maintains multiple, globally consistent checkpoints of the state of a shared memory multiprocessor (i.e., processors, memory, and coherence permissions), and it recovers to a pre-fault checkpoint of the system and re-executes if a fault is detected. *SafetyNet* e ...

**Keywords**: availability, shared memory, multiprocessor

8 A comparison of the object-oriented features of Ada 95 and Java

Benjamin M. Brosgol

November 1997 **Proceedings of the conference on TRI-Ada '97**

Full text available: pdf(2.41 MB) Additional Information: full citation, references, citings, index terms

9 Dynamically allocating processor resources between nearby and distant ILP ☐
Rajeev Balasubramonian, Sandhya Dwarkadas, David H. Albonesi
May 2001 **ACM SIGARCH Computer Architecture News , Proceedings of the 28th annual international symposium on Computer architecture**, Volume 29 Issue 2
Full text available: pdf(998.02 KB) Publisher Site Additional Information: full citation, abstract, references, citings, index terms

*Modern superscalar processors use wide instruction issue widths and out-of-order execution in order to increase instruction-level parallelism (ILP). Because instructions must be committed in order so as to guarantee precise exceptions, increasing ILP implies increasing the sizes of structures such as the register file, issue queue, and reorder buffer. Simultaneously, cycle time constraints limit the sizes of these structures, resulting in conflicting design requirements.*

*In ...*

10 Hardware Realization of a Java Virtual Machine for High Performance Multimedia Applications ☐
Mladen Berekovic, Helge Kloos, Peter Pirsch
August 1999 **Journal of VLSI Signal Processing Systems**, Volume 22 Issue 1
Full text available: Publisher Site Additional Information: full citation, abstract, index terms

This paper describes a new architecture for JAVA-based, interactive multimedia applications. A hardware implementation of a Java Virtual Machine (JVM) is proposed, which allows the direct execution of Java bytecode. In a single clock cycle, up to 3 bytecode instructions can be decoded and executed in parallel using a RISC pipeline. A splitable 64-bit ALU implementation addresses demanding processing requirements of typical multimedia signal processing schemes. The on-chip caches are ad ...

11 Safe Class and Data Evolution in Large and Long-Lived Java[tm] Applications ☐
Mikhail Dmitriev
August 2001 Technical Report, Sun Microsystems, Inc.
Full text available: pdf(876.82 KB) Additional Information: full citation, abstract

There is a growing class of applications implemented in object-oriented languages that are large and complex, that exploit object persistence, and need to run uninterrupted for long periods of time. Development and maintenance of such applications can present challenges in the following interrelated areas: consistent and scalable evolution of persistent data and code, optimal build management, and runtime changes to applications. The research presented in this thesis addresses the above issues. ...

12 Removing architectural bottlenecks to the scalability of speculative parallelization ☐
Milos Prvulovic, María Jesús Garzarán, Lawrence Rauchwerger, Josep Torrellas
May 2001 **ACM SIGARCH Computer Architecture News , Proceedings of the 28th annual international symposium on Computer architecture**, Volume 29 Issue 2
Full text available: pdf(1.13 MB) Publisher Site Additional Information: full citation, abstract, references, citings, index terms

Speculative thread-level parallelization is a promising way to speed up codes that compilers fail to parallelize. While several speculative parallelization schemes have been proposed for different machine sizes and types of codes, the results so far show that it is hard to deliver scalable speedups. Often, the problem is not true dependence violations, but sub-optimal architectural design. Consequently, we attempt to identify and eliminate major architectural bottlenecks that limit the scalab ...

13 Technical papers: mobile and distributed computing: A programming model and system support for disconnected-aware applications on resource-constrained devices ☐
Yaron Weinsberg, Israel Ben-Shaul
May 2002 **Proceedings of the 24th international conference on Software engineering**
Full text available: pdf(1.28 MB) Additional Information: full citation, abstract, references, index terms

The emergence of networked lightweight portable computing devices can potentially enable accessibility to a vast array of remote applications and data. In order to cope with shortage of local resources such as memory, CPU and bandwidth, such applications are typically designed as a thin-client thick-server applications. However, another highly desirable yet conflicting requirement is to support disconnected operation, due to the low quality and high cost of on-line connectivity. We present a nov ...

14 Technical correspondence: The HSSM macro-architecture, Virtual Machine and H languages ☐
Paul Damian Wells
April 2002 **ACM SIGPLAN Notices**, Volume 37 Issue 4
Full text available: pdf(752.57 KB) Additional Information: full citation, abstract, references

This paper is an introduction to the Hierarchical Simultaneous Set Membership (HSSM) macro-architecture project, the associated HSSM Virtual Machine (HVM) and "H" languages. The ultimate goal of this project is the automated synthesis and verification of large multithreaded software applications.

15 EMERALDS: a small-memory real-time microkernel ☐
Khawar M. Zuberi, Padmanabhan Pillai, Kang G. Shin
December 1999 **ACM SIGOPS Operating Systems Review , Proceedings of the seventeenth ACM symposium on Operating systems principles**, Volume 33 Issue 5
Full text available: pdf(1.59 MB) Additional Information: full citation, abstract, references, citings, index terms

EMERALDS (Extensible Microkernel for Embedded, ReAL-time, Distributed Systems) is a real-time microkernel designed for small-memory embedded applications. These applications must run on slow (15-25MHz) processors with just 32-128 kbytes of memory, either to keep production costs down in mass-produced systems or to keep weight and power consumption low. To be feasible for such applications, the OS must not only be small in size (less than 20 kbytes), but also have low-overhead kernel services. Un ...

16 Implementing jalapeño in Java ☐
Bowen Alpern, C. R. Attanasio, Anthony Cocchi, Derek Lieber, Stephen Smith, Ton Ngo, John J. Barton, Susan Flynn Hummel, Janice C. Sheperd, Mark Mergen
October 1999 **ACM SIGPLAN Notices , Proceedings of the 14th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**, Volume 34 Issue 10
Full text available: pdf(1.57 MB) Additional Information: full citation, abstract, references, citings, index terms

Jalapeño is a virtual machine for Java™ servers written in Java.A running Java program involves four layers of functionality: the user code, the virtual-machine, the operating system, and the hardware. By drawing the Java / non-Java boundary below the virtual machine rather than above it, Jalapeño reduces the boundary-crossing overhead and opens up more opportunities for optimization.To get Jalapeño started, a boot image of a ...

17 Java as a specification language for hardware-software systems ☐
Rachid Helaihel, Kunle Olukotun
November 1997 **Proceedings of the 1997 IEEE/ACM international conference on Computer-aided design**
Full text available: pdf(46.54 KB) Additional Information: full citation, abstract, references, citings, index

Publisher Site terms

The specification language is a critical component of the hardware-software co-design process since it is used for functional validation and as a starting point for hardware-software partitioning and co-synthesis. This paper pro poses the Java programming language as a specification language for hardware-software systems. Java has several characteristics that make it suitable for system specification. However, static control and dataflow analysis of Java programs is problematic because Java cla ...

**Keywords**: java, specification languages, hardware-software co-design

18 Superscalar architectures: Reducing the complexity of the register file in dynamic superscalar processors
Rajeev Balasubramonian, Sandhya Dwarkadas, David H. Albonesi
December 2001 **Proceedings of the 34th annual ACM/IEEE international symposium on Microarchitecture**
Full text available: pdf(1.34 MB) Additional Information: full citation, abstract, references, citings
Publisher Site

Dynamic superscalar processors execute multiple instructions out-of-order by looking for independent operations within a large window. The number of physical registers within the processor has a direct impact on the size of this window as most in-flight instructions require a new physical register at dispatch. A large multi-ported register file helps improve the instruction-level parallelism (ILP), but may have a detrimental effect on clock speed, especially in future wire-limited technologies. ...

19 Platforms: DFuse: a framework for distributed data fusion
Rajnish Kumar, Matthew Wolenetz, Bikash Agarwalla, JunSuk Shin, Phillip Hutto, Arnab Paul, Umakishore Ramachandran
November 2003 **Proceedings of the first international conference on Embedded networked sensor systems**
Full text available: pdf(541.24 KB) Additional Information: full citation, abstract, references, index terms

Simple in-network data aggregation (or fusion) techniques for sensor networks have been the focus of several recent research efforts, but they are insufficient to support advanced fusion applications. We extend these techniques to future sensor networks and ask two related questions: (a) what is the appropriate set of data fusion techniques, and (b) how do we dynamically assign aggregation roles to the nodes of a sensor network. We have developed an architectural framework, *DFuse*, for ans ...

**Keywords**: data fusion, energy awareness, in-network aggregation, middleware, platform, role assignment, sensor network

20 Compact garbage collection tables
David Tarditi
October 2000 **ACM SIGPLAN Notices , Proceedings of the second international symposium on Memory management**, Volume 36 Issue 1
Full text available: pdf(958.92 KB) Additional Information: full citation, abstract, index terms

Garbage collection tables for finding pointers on the stack can be represented in 20-25% of the space previously reported. Live pointer information is often the same at many call sites because there are few pointers live across most call sites. This allows live pointer information to be represented compactly by a small index into a table of descriptions of pointer locations. The mapping from program counter values to those small indexes can be represented compactly using several techniques. T ...

Results 1 - 20 of 60 Result page: **1** 2 3 4 next

US-PAT-NO: 6047362

DOCUMENT-IDENTIFIER: US 6047362 A

TITLE: Delayed removal of address mapping for terminated processes

DATE-ISSUED: April 4, 2000

INVENTOR-INFORMATION:

| NAME | CITY | STATE | ZIP CODE | COUNTRY |
|---|---|---|---|---|
| Zucker; J. Steven | Manhattan Beach | CA | N/A | N/A |

US-CL-CURRENT: 711/203, 711/133 , 711/134 , 711/156 , 711/159 , 711/206

ABSTRACT:

An application binary interface includes linkage structures for interfacing a binary application program to a digital computer. Virtual address spaces are allocated for processes respectively. Page table entries for translation of the virtual address spaces into physical addresses are not removed as processes terminate, but only after all virtual address spaces have been allocated.

32 Claims, 25 Drawing figures

Exemplary Claim Number: 1

Number of Drawing Sheets: 11

---------- KWIC ---------

Detailed Description Text - DETX (142):
As illustrated in FIG. 20, the allocation table 154 comprises a linked list of entries, which are kept sorted by VSID range and coalesced. Each entry comprises a number (START) of the first VSID range in a block of contiguous unallocated VSID ranges, a number (FREE) of unallocated VSID ranges in the contiguous block starting with the value **START and a pointer** (LINK) to the next entry. The entries need not be stored contiguously in memory, but can reside at any addresses since they are linked together by the pointers LINK.

Detailed Description Text - DETX (153):
As illustrated in the flowchart of FIG. 21, the unmapping control unit 152 first acquires a mutual exclusion (**mutex**) lock from the operating system 48 that prevents any other operation from accessing the allocation table 154 and the usage table 156. The unmapping control unit 152 then scans all PTEs in the page table in sequential order.

Detailed Description Text - DETX (158):
The unit 150 then locates the next free VSID range in the usage table 156, creates a second entry for the allocation table 150 that is pointed to from the first entry, and continues this process until the allocation table 150 has been reconstructed by creating an entry for each group of contiguous free VSID ranges. After the allocation table 150 has been reconstructed, the **mutex** lock is released and the mapping control unit 150 is allowed to continue allocating new VSID ranges for mapping by the memory management unit 18.

US-PAT-NO: 5448706

DOCUMENT-IDENTIFIER: US 5448706 A

TITLE: Address generator for multi-channel circular-buffer style processing

DATE-ISSUED: September 5, 1995

INVENTOR-INFORMATION:

| NAME<br>COUNTRY | CITY | STATE | ZIP CODE | |
|---|---|---|---|---|
| Fleming; Michael E. | Vancouver | WA | N/A | N/A |
| Anderson; Eric C. | Vancouver | WA | N/A | N/A |

US-CL-CURRENT: 711/217, 708/421 , 711/200

ABSTRACT:

A one-chip address generator for producing a sequence of address signals for application to a memory containing a plurality of circular buffers. The address generator chip is capable of processing service requests from a plurality of channels on a prioritized basis. Service requests can arrive asynchronously at different rates. A channel-specific length or overlap value can be assigned to each servicing of a request. A seamless pipeline structure is provided for processing the service requests of subsequent channels immediately after completion of service for a first requesting channel.

25 Claims, 11 Drawing figures

Exemplary Claim Number: 1

Number of Drawing Sheets: 11

---------- KWIC ---------

Detailed Description Text - DETX (118):

In summary, the pointer update operation consumes three clock cycles as shown at the bottom of the state flow diagram of FIG. 7. The first cycle is used for **reading out the current start pointer** SP from register file 265 and making the first pass through subtracter/adder 268. Subtracter/adder 268 is in the subtract mode during the first pass. The second cycle is used for making the second pass through subtracter/adder 268. Subtracter/adder 268 is in the add mode during the second pass. The third update cycle is used for loading the new SP value into register file 265. Updating of the start-pointers register file 265 occurs concurrently with state-variable down-counting in down counter 289. Update cycle one ,occurs concurrently with a state defined in FIG. 7 as SV=L'.sub.Cn. Update cycle two occurs concurrently with a state defined as SV=L'.sub.Cn -1. Update cycle three occurs concurrently with a state defined as SV=L'.sub.Cn -2. (For L'.sub.Cn =3, state SV=L'.sub.Cn -1 becomes the same as state SV=2 and state SV=L'.sub.Cn -2 becomes the same as state SV=1.)

Claims Text - CLTX (61):

(c) in response to the channel number signal, reading out from the second memory the start pointer of the identified circular buffer and transmitting the

**read-out start pointer** to a sequencer;

Claims Text - CLTX (62):
(d) generating an updated **start pointer that is derived from the read-out start pointer and, after the read-out start pointer** is transmitted to the sequencer, writing the updated start pointer into the second memory at the location previously occupied by the **read-out start pointer**;

Claims Text - CLTX (63):
(e) combining the **read-out start pointer** with the generated channel number signal to thereby produce a first absolute address signal; and

Claims Text - CLTX (75):
(e.3) in response to said first and second determinations, defining a first field within the first absolute address signal corresponding to the **read-out start pointer** and a second field within the first absolute address signal corresponding to the generated channel number signal.